Serverless computing is a new paradigm for building distributed applications. A serverless application relies entirely on a platform for executing and scaling functions. Serverless computing may support Fog Computing in practice. In Fog computing, applications can improve latency, responsiveness, and bandwidth by offloading work earlier in the processing pipeline. Currently, no serverless framework allowing the utilization of the different heterogeneous resources in Fog exists.

To further the development of a Serverless-Fog-Environment, we want to address the following challenges through Master Theses:

1. FaaS extension for long running connections

Not all types of applications are currently possible within the serverless application model. For instance, “Web applications often issue [...] long-lived RPC calls as a part of a long polling technique; [...].” A technique like this is not possible in the current FaaS ecosystem. Can the serverless ecosystem be extended with a middleware that can handle such long-lived connections?

**Topic & Goal:** The objective is to select either OpenLambda or OpenWisk and implement a long-lived connection middleware. The result needs to be especially evaluated regarding provider costs.

2. Deployment of Function Runtimes in heterogeneous environments

Container engines and systems like Firecracker are currently used to manage function-runtimes. However, these systems are not yet designed to work across different system architectures. Is it possible to extend an existing container management framework to allow the seamless deployment of function-runtimes on different system architectures?

**Topic & Goal:** For this thesis, the objective is to extend an existing virtualization framework with the ability to deploy a serverless runtime on different system architectures transparently.

3. Workload Predictor for FaaS Runtimes

Serverless applications suffer from a cold start problem, where it takes time to start up an application instance if it has not been used for a long time. There are different techniques to reduce this cold-start problem, but all come with a cost. Can access patterns of a serverless application be used to anticipate requests in order to reduce the cold start latency?

**Topic & Goal:** The objective is to build a prototype that predicts traffic patterns of a serverless application and use that to warm up function container.

Interested in other Serverless Topics? Let me know!

**Contact:** Sebastian Werner
werner@tu-berlin.de